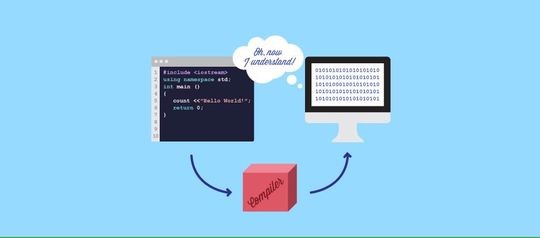


EMSI 3 – 4°IIR2 – COMPILATION – Prof. M.D. RAHMANI

**EMSI – IIR – INFORMATIQUE**

**RESOLUTION DE L’EXERCICE 1 :**

Youssef **MAHTAT**

**RÉALISÉ PAR :**

**DEVOIR COMPILATION**

**YACC in Bison**

**&**

**RegEx in Flex**

**Réalisation des Analyseurs LEXICALS et SYNTAXIQUE**

**→** Fichier analyseur.l :

%{

#include <stdio.h>

#include <stdlib.h>

#include "PrintSymbolTableLib.cpp" // Bibliothèque d'affichage réaliser pour l'analyseur

// **addOutput(**char\* lexeme, char\* classUnit, int index**)** : ajout un lexème analysé à l'affichage final (résultat de l'analyse)

// **addErrorOutput(char\* lexeme, int index)** : ajout une erreur trouvé à l'affichage final (résultat de l'analyse)

// **printOutput()** : affiche le résultat de l'analyse

int index = 1; // indice du 1er mot, puis sera incrémenté à chaque lexème trouvé !

%}

delim [\t ]

bl {delim}+

lettre [a-zA-Z]

chiffre [0-9]

id {lettre}({lettre}|{chiffre})+

signe (\+|\-)?

nbr {chiffre}+(.{chiffre}+)?((e|E){signe}{chiffre}+)?

%%

\n {/\* NE RIEN FAIRE \*/}

{bl} {/\* IGNORE \*/}

"ENTIER" {addOutput(yytext,"MOT-CLE",index);index++;}

"REEL" {addOutput(yytext,"MOT-CLE",index);index++;}

{id} {addOutput(yytext,"IDENTIFICATEUR",index);index++;}

{nbr} {addOutput(yytext,"NOMBRE",index);index++;}

"," {addOutput(yytext,"VIRGULE",index);index++;}

";" {addOutput(yytext,"POINT-VIRGULE",index);index++;}

"=" {addOutput(yytext,"AFFECTATION",index);index++;}

"+" {addOutput(yytext,"ADDITION",index);index++;}

"-" {addOutput(yytext,"SOUSTRACTION",index);index++;}

"\*" {addOutput(yytext,"MULTIPLICATION",index);index++;}

"/" {addOutput(yytext,"DIVISION",index);index++;}

"(" {addOutput(yytext,"PARENTHESE OUVRANTE",index);index++;}

")" {addOutput(yytext,"PARENTHESE FERMANTE",index);index++;}

"$" {addOutput(yytext,"FIN",index);index++;printOutput(); exit(0);}

. {addErrorOutput(yytext,index);index++;}

%%

int main()

{

    printf("\nExpressions a analyser (terminees par $) : \n");

    yylex();

    return 0;

}

int yywrap()

{

    // le delimiteur de la lecture

    return 1;

}

**→** Fichier PrintSymbolTableLib.cpp:

#include <stdio.h>

#include <stdlib.h>

#include <iostream>

#include <string>

using namespace std;

string output = ""; // variable qui contiendra le résultat de l'analyse

char temp[100]; // variable tampon, pour convertir un int to string

// Ajout une ligne à la variable output, qui contient le (lexeme, unité lexical, indice) :

void addOutput(char\* lexeme,char\* classUnit, int index)

{

// Preparation de la ligne, et l'ajout partie par partie !

output += "LEXEME : " ;

output += lexeme;

output += " | lexicalUnit : ";

output += classUnit;

output += " | index ; ";

sprintf(temp,"%d",index);

output += temp;

output += " ;\n";

}

// Ajout une ligne d'erreur à la variable output, pour un mot qui n'est pas dans le lexique

void addErrorOutput(char\* lexeme, int index)

{

// Preparation de la ligne d'erreur, et l'ajout partie par partie !

output += "motERREUR : " ;

output += lexeme;

output += " | NON INCLU DANS LE LEXIQUE";

output += " | index ; ";

sprintf(temp,"%d",index);

output += temp;

output += " ;\n";

}

// Affiche la variable output, le résultat de l'analyse

void printOutput()

{

cout << output;

}

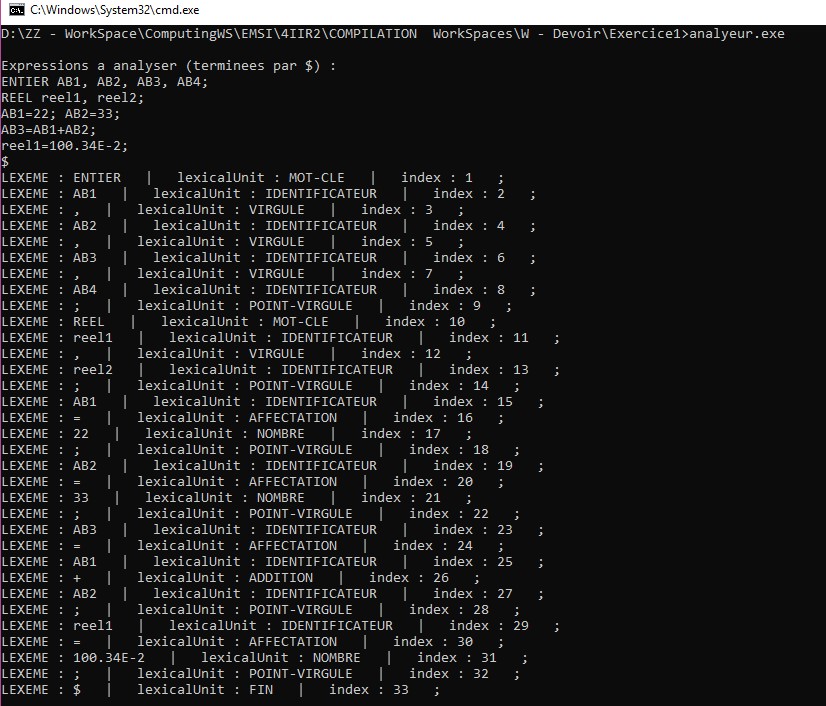
**RmQ – Pour Compiler et Exécuter l’analyseur :**

cmd>> flex analyseur.l

cmd>> g++ lex.yy.c -o analyseur

cmd>> analyseur.exe

**→** Exemple d’exécution :



**RESOLUTION DE L’EXERCICE 2 :**

* **CONCEPTION & ETUDE INITIALE :**

**→ Grammaire Régulière :**

**La syntaxe de la calculatrice scientifique suit la grammaire suivante :**

SENS DE LA PRIORITE

**R → E $**

**E → E + T | E – T | T**

**T → T \* F | T / F | F**

**F → nbr | ( E ) | -F | +F | F ^ F | cosinus F | sinus F | tan F | exp F**

**| log F | valeur\_absolu F | racine\_carre F**

**→ RmQ sur l’ANALYSE LEXICAL :**

**Un nombre réel est validé par l’expression :**

**(\+|\-)? [0-9]+(\.[0-9]+)?((e|E)(\+|\-)?[0-9]+)?**

* **IMPLANTATION DE LA CALCULATRICE EN FLEX ET BISON (ET C/C++) :**

**→ Fichier golobal.h :**

#define YYSTYPE double

extern YYSTYPE yylval;

**→ Fichier calculatrice.y :**

%{

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include "global.h"

int yyerror(char\* s);

int yylex();

%}

%token NOMBRE PO PF FIN

%left PLUS MOINS

%left FOIS DIV

%nonassoc COSINUS SINUS RACINE ABS LOGARITHME TANGENTE EXPONENTIELLE

%right PUISSANCE

%start R

%%

R : E FIN {printf("Resultat = %f",$1);return 0;};

E : E PLUS T {$$=$1+$3;}

  | E MOINS T {$$=$1-$3;}

  | T {$$=$1;}

;

T : T FOIS F {$$=$1\*$3;}

  | T DIV F {if ($3 == 0) {printf ("division par zéro interdite");exit(1);} else{ $$ = $1 / $3;}}

  | F

;

F : NOMBRE

| PO E PF {$$=$2;}

  | MOINS F %prec NEG {$$= -$2;}

  | PLUS F %prec POS {$$= +$2;}

  | F PUISSANCE F {$$ = pow($1,$3);}

  | COSINUS F {$$ = cos($2);}

  | SINUS F {$$ = sin($2);}

  | TANGENTE F {$$ = tan($2);}

  | LOGARITHME F {$$ = log($2);}

  | EXPONENTIELLE F {$$ = exp($2);}

  | ABS F {$$ = fabs($2);}

  | RACINE F {$$ = sqrt($2);}

;

%%

int yyerror(char \*s)

{

  printf("%s",s);return 1;

}

int main()

{

  printf("Donnez une expression : ");

  yyparse();

  return 0;

}

**→ Fichier calculatrice.l :**

%{

#include <math.h>

#include <stdlib.h>

#include "global.h"

#include "calculatrice.tab.h"

%}

nombre [0-9]+(\.[0-9]+)?((e|E)(\+|\-)?[0-9]+)?

%%

{nombre} {yylval=atof(yytext); return NOMBRE;}

"\n" {return FIN;}

"+" {return PLUS;}

"-" {return MOINS;}

"\*"      {return FOIS;}

"/"      {return DIV;}

"(" {return PO;}

")" {return PF;}

"^"      {return PUISSANCE;}

"cosinus"        {return COSINUS;}

"sinus"      {return SINUS;}

"log"        {return LOGARITHME;}

"valeur\_absolu" {return ABS;}

"racine\_carre" {return RACINE;}

"tangente" {return TANGENTE;}

"exp" {return EXPONENTIELLE;}

. { }

%%

int yywrap(void)

{

    return 1;

}

**RmQ – Pour Compiler et Exécuter les analyseurs (CALCULATRICE SCIENTIFIQUE) :**

Cmd>> bison calculatrice.y

cmd>> flex calculatrice.l

cmd>> gcc lex.yy.c calculatrice.tab.c -o calculatrice

cmd>> calculatrice.exe

* **Exemple d’exécution :**

